SANDIA REPORT

SAND97–2340 • UC–706
Unlimited Release
Printed October 1997

Measuring Worst-Case Errors in a Robot Workcell

Ronald W. Simon, Randy C. Brost, Deepesh K. Kholwadwala

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
Office of Scientific and Technical Information
PO Box 62
Oak Ridge, TN 37831

Prices available from (615) 576-8401, FTS 626-8401

Available to the public from
National Technical Information Service
US Department of Commerce
5285 Port Royal Rd
Springfield, VA 22161

NTIS price codes
Printed copy: A08
Microfiche copy: A01

SAND-97-2340
Unlimited Release
Printed October 1997

Distribution
Category UC-706

Measuring Worst-Case Errors in a Robot Workcell

Ronald W. Simon
Randy C. Brost
Deepesh K. Kholwadwala
Intelligent Systems and Robotics Center
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-5800

Abstract

Errors in model parameters, sensing, and control are inevitably present in real robot systems. These errors must be considered in order to automatically plan robust solutions to many manipulation tasks. Lozano-Pérez, Mason, and Taylor proposed a formal method for synthesizing robust actions in the presence of uncertainty [7]; this method has been extended by several subsequent researchers. All of these results presume the existence of worst-case error bounds that describe the maximum possible deviation between the robot's model of the world and reality. This paper examines the problem of measuring these error bounds for a real robot workcell. These measurements are difficult, because of the desire to completely contain all possible deviations while avoiding bounds that are overly conservative. We present a detailed description of a series of experiments that characterize and quantify the possible errors in visual sensing and motion control for a robot workcell equipped with standard industrial robot hardware. In addition to providing a means for measuring these specific errors, our experiments shed light on the general problem of measuring worst-case errors.

Contents

1 Introduction

No real robot system operates with perfect sensory information or control capabilities. Consequently, real robot sensory data is an imperfect measurement of the world, and real robot motions only approximate the intended motions. These factors may cause robot actions that will succeed in an ideal world to fail during execution in the real world.

This reality is one of the key reasons that automatic planning and execution systems are not currently employed to solve tasks with high precision requirements. For example, assembly tasks often require insertions that have tolerances that are much tighter than the sensing and motion control capabilities of current industrial robots. Thus, an automatic task planning system cannot simply command a robot motion to move the parts into their desired positions; such a motion will inevitably lead to collisions that do not occur in a perfectly executed motion.

Compliant motion strategies are often used to successfully accomplish insertion tasks. Part features such as chamfers guide the parts into the desired locations, effectively removing errors. Similar strategies may be employed to improve the reliability of part acquisition tasks. To date these strategies are designed manually, since automatic planning systems do not exist that can automatically synthesize compliant motion strategies for real robot tasks.

Lozano-Pérez, Mason, and Taylor [7] have proposed a method for synthesizing robust compliant motion strategies. In brief, this method assumes a worst-case model of robot errors, and performs an analysis of the task geometry and mechanics to identify compliant motions that will accomplish the task goal in the presence of errors. Lozano-Pérez, Mason, and Taylor explain their approach using an abstract model of robot action; if we can implement a planner that performs this analysis for real tasks and provide the necessary input data, then automatic planning of robust compliant motions can become a reality.

The development of such a planner has been a primary focus of the Sandia Manipulation Laboratory for several years. Our work toward an automatic planning system will be described in a forthcoming paper. In this paper, we focus on the input part of the problem: How can we obtain the required worst-case error bounds?

Figure 1 shows a simple example that motivates this problem. In this scenario, a robot is attempting to insert a peg into a hole. The robot has a model of the hole's location, expressed as (x, y) coordinates. The robot also has a model of the peg's position relative to its gripper. The robot inserts the peg by calculating the arm position that will place the peg directly above the hole, moving there, and lowering the peg into the hole.

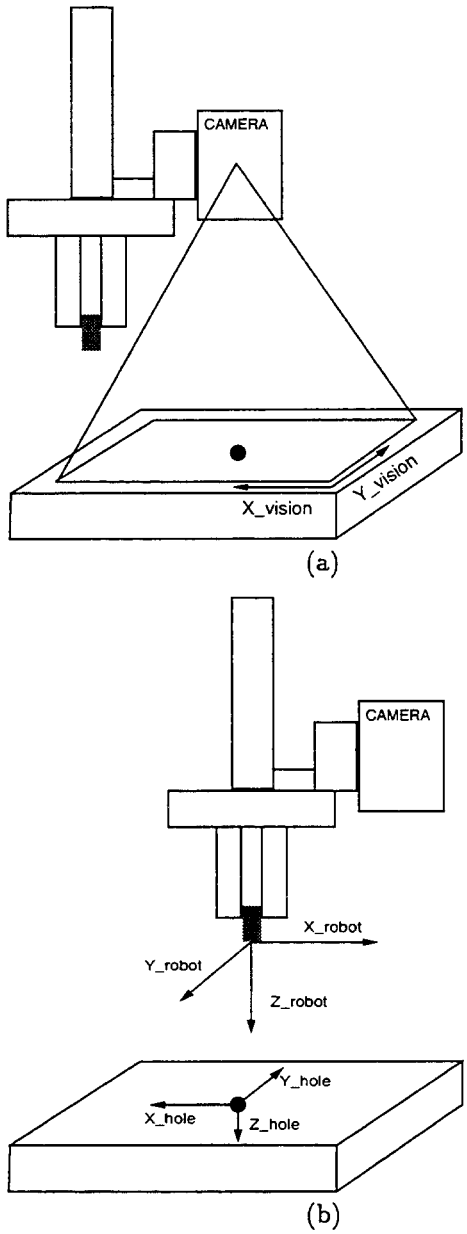


Figure 1: (a) The robot must locate the hole in vision coordinates. (b) The robot must insert the peg into the hole.

In an ideal world, the true positions of the peg and hole will exactly correspond to their modeled positions, and the robot's motion will exactly track the commanded motion. Under these conditions, the insertion will always succeed. In the real world, errors will arise that cause the true positions to deviate from the modeled positions, and the robot motion to deviate from the intended motion. These errors may cause the in-

sertion to fail.

We can determine whether failure is possible by performing a worst-case error analysis. We begin by observing that the allowable error in the peg's position before insertion is $\epsilon_{max} = r_{hole} - r_{peg}$, which is the peg/hole clearance. If the distance between the (x, y) coordinates of the center of the peg and the center of the hole is less than ϵ_{max}, then the peg will slip into the hole. The maximum possible deviation is obtained by $\epsilon_{total} = \epsilon_{hole} + \epsilon_{grasp} + \epsilon_{motion}$, where ϵ_{hole}, ϵ_{grasp}, and ϵ_{motion} describe the worst-case errors in the hole position, the peg position relative to the gripper, and the robot's motion accuracy. If $\epsilon_{total} < \epsilon_{max}$, then the insertion will be reliable.

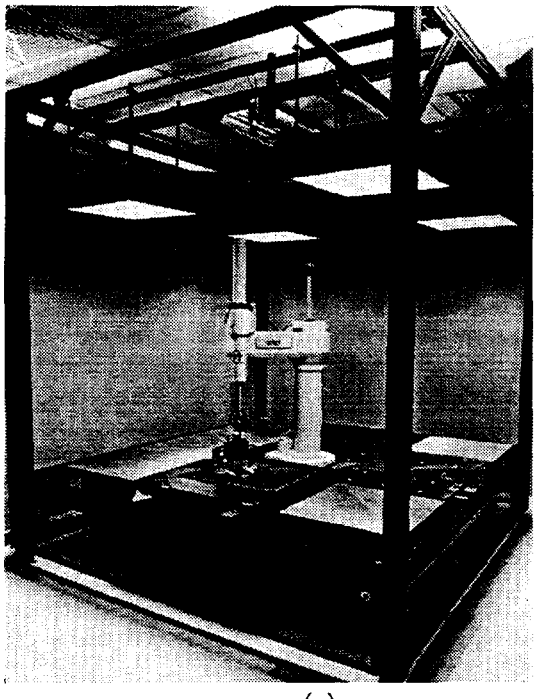
This example shows how worst-case error bounds may be used to evaluate the reliability of a proposed robot action. The Lozano-Pérez, Mason, and Taylor method generalizes this technique to problems with more complicated task geometry and mechanics, considering additional sources of error. Their work has been extended and refined by subsequent authors, resulting in several prototype automatic planning systems [5, 4, 6, 1].

One assumption underlying all of these research results is that worst-case error bounds such as ϵ_{hole}, ϵ_{peg}, and ϵ_{motion} may be obtained and provided as input to the planner. How can these error bounds be measured?

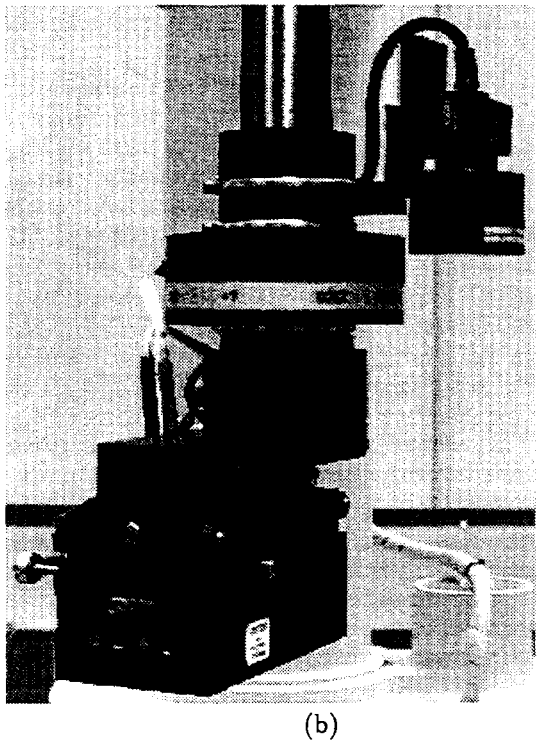
This paper describes a series of experiments performed in the Sandia Manipulation Laboratory to characterize the worst-case errors in visual sensing and motion accuracy for the laboratory workcell. These experiments served to measure the inherent error characteristics of the workcell for later use in task analysis experiments, as well as providing insight into the general problem of estimating worst-case error bounds.

The workcell is shown in Figure 2. An Adept One manipulator is mounted on a rigid steel base which is welded to the worktable. This worktable is roughly two meters square, and is precision-ground so that the work surface is flat and level to within ± 0.1mm. Tooling balls are mounted on the table to facilitate repeatable placement of experimental modules, attached to standard mounting plates. Surrounding the worktable is a truss structure designed to support lights, lexan safety barriers, and coarse-view overhead cameras. The lights are designed and located to provide even illumination of the table surface; measurements indicate that light intensity varies by only 5% over the table work area. The robot is equipped with an end-effector which includes a gripper, compliant wrist, force sensor, and close-view camera. Visual processing is performed using standard vision analysis procedures included in the Adept controller (see Section 3.1). In the experiments we describe here, the force sensor and compliant wrist are not used.

This workcell was designed to perform a series of ex-



(a)



(b)

Figure 2: (a) The workcell. (b) Close-up of the ro end-effector.

periments measuring the robustness of automatic planned manipulation actions. These experiments have the same basic format: Given a desired action

test, the robot visually measures the current location of an object to be manipulated, calculates and moves to the desired starting position relative to the object, executes the action, and checks the result using vision and other sensors. This process is repeated autonomously over thousands of trials, thus obtaining action reliability measurements over a range of varying initial conditions. A common theme in all of these experiments is a basic strategy for initiating the action: Visually measure the object's location in the world, compute the robot position to achieve the desired relative position, and move there.

These experiments require worst case error bounds ϵ_{vision} and ϵ_{motion} describing the maximum possible error in the measured object location, and the arm's position accuracy. Our work currently focuses on planar models of action, where points in space are described by (x, y, θ) coordinates. Thus to characterize the error properties of the basic measure-calculate-move action setup strategy, we need four error bounds:

$\epsilon_{\text{vision}_{xy}}$
The maximum possible distance between the measured (x, y) object position and its true position, relative to the camera.

$\epsilon_{\text{vision}_\theta}$
The maximum angular deviation between the measured object orientation θ and its true orientation, relative to the camera.

$\epsilon_{\text{motion}_{xy}}$
The maximum possible distance between the true (x, y) position of the robot's quill center at the end of a motion, and the intended position. The goal position may be an arbitrary input (x, y) point, and is not taught. Thus, $\epsilon_{\text{motion}_{xy}}$ is a measure of the robot's accuracy, not its repeatability.

$\epsilon_{\text{motion}_\theta}$
The maximum angular deviation between the true θ orientation of the robot's tool flange at the end of a motion, and the intended orientation. Again, $\epsilon_{\text{motion}_\theta}$ is a measure of the robot's accuracy, not its repeatability.

In order to obtain consistent visual position measurements over a broad class of task objects, we employ a uniform object position measurement procedure. We paint two white circular dots on an object, and paint the object and background flat black to eliminate other features and shadows. If the dots have different diameters, then the object position and orientation are obtained directly from the measured dot positions. Thus $\epsilon_{\text{vision}_{xy}}$ is equal to the maximum possible error in locating a dot, and $\epsilon_{\text{vision}_\theta} = \tan^{-1}(2\epsilon_{\text{vision}_{xy}}/d)$, where d is the distance between dot centers. The motion errors $\epsilon_{\text{motion}_{xy}}$ and $\epsilon_{\text{motion}_\theta}$ are intrinsic properties of the robot arm and its controller. Therefore we need to measure $\epsilon_{\text{vision}_{xy}}$, $\epsilon_{\text{motion}_{xy}}$, and $\epsilon_{\text{motion}_\theta}$ for our robot system.

Measuring these values is tricky. To support the goals of compliant motion planning using worst-case analysis methods, we must assure that no error can ever exceed its associated ϵ bound, while also avoiding ϵ values that are overly conservative. These competing requirements prevent the blind application of conventional RMS or 6σ error modeling techniques, at least for the cases we studied.

The remainder of this paper will explain how we measured and verified these error bounds for the Sandia Manipulation Laboratory workcell. Section 2 will compare our results to past work in robot calibration. Section 3 will address $\epsilon_{\text{vision}_{xy}}$, and Section 4 will address $\epsilon_{\text{motion}_{xy}}$ and $\epsilon_{\text{motion}_\theta}$. Section 6 will discuss some of the lessons learned during this exercise. In brief, we view our results as positive because we successfully measured the required worst-case error bounds for a complex robot system, but worrisome because of the effort required to obtain these measurements. Our final results for the error bounds of the workcell are:

$$\epsilon_{\text{vision}_{xy}} = 0.119\text{mm}$$

$$\epsilon_{\text{vision}_\theta} = \tan^{-1}\left(\frac{2\epsilon_{\text{vision}_{xy}}}{d}\right)$$

$$\epsilon_{\text{motion}_{xy}} = 0.13\text{mm}$$

$$\epsilon_{\text{motion}_\theta} = 0.20^\circ$$

where d is the distance between the dots on the object being measured. These values of $\epsilon_{\text{vision}_{xy}}$ reflect the results of applying correction terms to eliminate substantial lens distortion effects; without these corrections, $\epsilon_{\text{vision}_{xy}}$ would be 6mm.

2 Previous Work

Much work has been done to evaluate the capability of vision systems to locate objects with a defined precision. The field of photogrammetry was an early driver [2, 3], as increasingly higher levels of accuracy were sought. Often a model was developed to relate camera parameters to resultant object displacements from ideal locations within a scene. The complexity of the models did not always correlate to improved camera characterization, and limited the range of applications addressed. Simulation was often used to provide testimonial of the improvements in performance, and results may or may not account for noise or lens distortion. Several computationally intense algorithms have been developed to provide subpixel registration within a scene (e.g, [11]). Time constraints limited the applicability of many of these routines. In 1985 Roger Tsai [13] proposed a method for a camera calibration using standard TV cameras and lenses, and in 1987

Peter Seitz [8] proposed an algorithm to provide better accuracy for these cameras. Tsai's procedure has the advantage of only needing to be performed once for a given camera, but required a strong model of lens distortion and ignored the effect of noise. Seitz's work required symmetrical distortion of the lens, the use of symmetrical filtering, and also ignored the contribution of system noise. The previous results failed to address all contributors to video system distortion. Our work devises a method to characterize worst-case errors that accounts for all of the error contributions.

We chose to analyze the relationship between the camera and the image plane as an arbitrary mapping of smooth distortion values. This allowed us to use a very weak model of the camera parameters at the expense of system memory used for storing a lookup table. Once the camera is corrected, it is used to provide a mapping of displacements of the robotic arm in the workcell. We also chose to use lookup tables (with inherent memory requirements) instead of characterizing and correcting a strong model of the robot kinematics.

In 1994 Shah and Aggaral [9] published a paper documenting a similiar process in their work for correcting a fish-eye lens. The approach taken by Smagt, Groen, and Krose [10] in their work at the University of Amsterdam in 1993 relied upon vision systems and a nueral control algorithm to correct the kinematics of the robot in an adaptive control process. A drawback to this approach is the computational complexity which requires specialized hardware to provide real time processing.

Adept Technology has a commercial product that is essentially the same as our position table lookup algorithm. Their product may be used to map an area as large as ours (by overlapping smaller area maps) with some increased complexity, however it does deliver better accuracy. The Adept product requires end of arm tooling (a light point source) attached to the robot. Points on a glass grid are then visited and compared to the commanded point. Errors are recorded in a lookup table and movements to these points are corrected as they are made. Our method does not require additional hardware if the camera is required during experiments. Another advantage of our method is that all measurements are made with the experiment tooling in place, and drift due to temperature variations and wear on the manipulator may be corrected on-line. These corrections are not currently implemented.

3 Measuring ϵ_{vision}

3.1 Dot Location

As explained above, we are interested in characterizing the error in the robot's ability to measure the (x, y) position of a high-contrast circular dot in its workspace, using the close-view camera attached to its end-effector. To focus on vision-related effects, we will express this error in terms of the camera coordinate system, and not include possible errors in the robot's positioning of the camera. These errors will be addressed separately in Section 4.

There are several possible methods for locating circular dots in an image; the Adept vision system provides at least three: general blob analysis, circular arc fitting, and linear rulers. General blob analysis is performed by classifying pixels into groups by intensity levels and finding the centroid of the pixels classified as members of a dot. The circular arc fitting method identifies pixels corresponding to the boundary of the circular dot, and fitting a circle to these points. The ruler-based method is performed by using linear edge-detecting "rulers" placed vertically and horizontally, to measure the diameter of the dot and thus its center point. See Figure 3.

These methods vary in their robustness. Recall that in our workcell, the lights were designed to provide even illumination of the work area. This even illumination is corrupted by the presence of the robot, which obstructs a few of the light sources. Because light comes from several different directions, the resulting shadows have fairly low contrast; however, this does cause variations in the overall scene intensity related to the robot's configuration. These variations can adversely affect the general blob analysis methods, because the pixel classification schemes they employ are sensitive to intensity changes. In contrast, the linear ruler method remains robust in the face of smooth intensity changes, since the dot boundary still produces a high-contrast edge even when the total illumination varies.

The dot-finding methods also vary in their sensitivity to dot size. As the size of the dot increases, the number of pixels covered by the dot also increases. The resulting increase in information can improve the effective resolution of both the blob analysis and arc fitting methods, since more pixels contribute to the statistically averaged estimate of the dot center or boundary. Conversely, resolution degrades as the dot size shrinks. Since we expect to manipulate small objects in some of our experiments, we would prefer to use a method whose accuracy does not degrade with dot size.

For these reasons, we chose to employ the ruler-based method of finding dots. This is accomplished by starting with a nominal (x, y) dot location that is known either from the expected dot position, or preliminary blob analysis. A horizontal ruler is used to detect edges on either side of this center point; the resulting edge-crossing x-values are used to refine the estimate of the dot center x value. A vertical ruler is then placed at this refined x-value, and the edge-crossing y-values are used to determine a refined dot center y value. Since the original horizontal ruler was likely placed above or below the dot centerline, the crossing angle of the dot edges was likely not perpendicular. To avoid a loss of accuracy in the dot's measured x position, a second horizontal ruler is placed, now using the refined x and

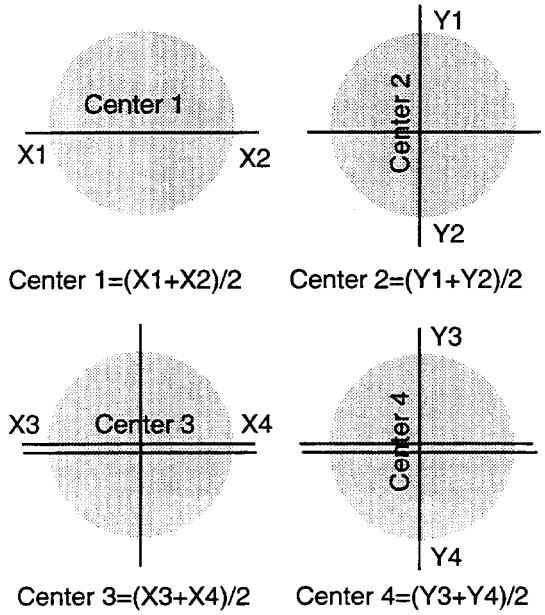


Figure 3: The dot location method. This procedure was tested with up to twelve lines; convergence after four lines made further iterations redundant.

y coordinates obtained thus far. This yields a further refined x' value. This process is illustrated in Figure 3.

Further refinements are certainly possible, yielding a series of incrementally improved $y', x'', y'', \ldots$ values. Testing revealed that values reliably converged by x'', so we adopted a canonical strategy of stopping refinement after x'' was calculated. The resulting dot-finding strategy thus requires four ruler edge-detection operations per dot, and some simple arithmetic.

3.2 Error Model

After selecting a concrete dot-finding procedure, we can proceed to characterize its error properties. This immediately raises a fundamental question: How do we measure worst-case errors?

This question raises a number of issues, not the least of which is what we mean by "worst-case." One could argue, given an adversary allowed to impose any catastrophic event, that it is impossible to bound the set of possible behaviors of a system. This includes turning off all the lights, unplugging the camera, painting defects on the dot, etc. In the worst case, very large errors will result if certain critical logic circuits fail in just the right way, returning values such as 3.45×10^{67}mm. In order to cover all of these scenarios, we need to select an ϵ_{vision} equal to the largest number that can be represented by the Adept processor.

This doesn't seem to be very useful. Instead, we will assume that such catastrophes are avoided, and the returned (x, y) coordinates represent the result of a normal analysis of a proper dot image. Under this assumption, how much error can there be between the measured and true dot positions?

One possibility would be to develop some means of comparing measured values against ground truth, take a large number of such measurements, and then calculate the standard deviation σ of the resulting error values. We could then take $\epsilon_{\text{vision}_{xy}} = 6\sigma$, and assert that error values outside this range are practically impossible. This approach is somewhat unsatisfying. The method is perhaps reasonable when error distributions are Gaussian, but so far we have no justification for assuming that our vision errors are Gaussian in nature. If the errors are not Gaussian, then our resulting error bound $\epsilon_{\text{vision}_{xy}}$ may be incorrect. If we take a large enough sample of measurements to capture all of the error phenomena that may arise, then $\epsilon_{\text{vision}_{xy}}$ will generally be too large if the error distribution is not truly Gaussian. As we shall see later, blindly applying a Gaussian model to raw vision error data will yield an $\epsilon_{\text{vision}_{xy}}$ that is very conservative.

A better approach is to consider the physical process employed by the sensors, and identify particular sources of errors. This allows us to measure each error component individually, through experiments designed to isolate the source of error. In visual sensing, the basic process is that light reflects off the scene, passes through a lens and creates an image on a camera CCD array. The image is converted to an electronic signal and sent to the vision processor. Sampling the signal at intervals which correlate to the size of the frame store buffer provides a new representation of the camera scene. Several image processing computations may be performed on the frame buffer to analyze this representation. This view of the visual sensing process illuminates several possible sources for error: The transmission of the light through the lens, the capture of the light by the CCD array, the conversion and transmission of the resulting CCD image intensities to the vision processor, the sampling routine used to fill the frame buffer, and the image analysis computation. We assume that there will be error contributions from each of these sources and group them as follows: the lens, the CCD array/video processing, and the electronic transmission of the image. If each of these processes has an associated error bound ϵ, then

$$\epsilon_{\text{vision}_{xy}} = \epsilon_{\text{lens}} + \epsilon_{\text{pixel}} + \epsilon_{\text{noise}}$$

ϵ_{lens} is the fixed distortion due to imperfections in the lens, ϵ_{pixel} is the fixed error resulting from the CCD characteristics, sampling rate variations, and video processing limitations, and ϵ_{noise} represents the lumped effect of noise in the electronic image transmission as well as any noise that may be present due to lens and CCD properties.

In the sections that follow, we will develop a model of the errors giving rise to ϵ_{lens}, ϵ_{pixel}, and ϵ_{noise}, design an experiment to characterize each error, and present

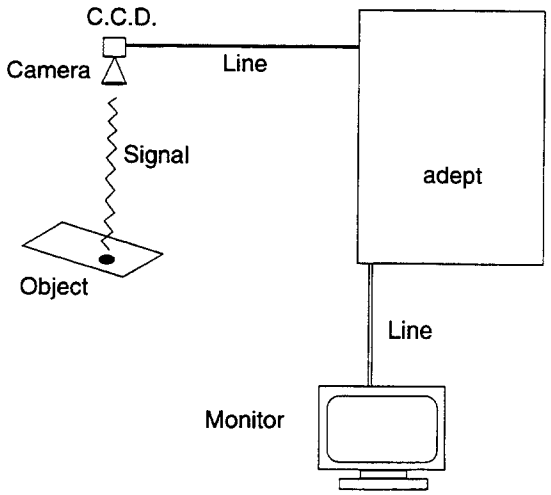


Figure 4: Pictoral representation of system contributers to optical/electronic noise in video proccessing. Noise variations affect the image plane location of dot centers.

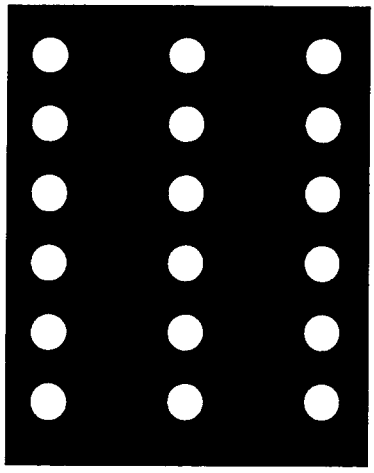


Figure 5: 3 by 6 grid used to characterize noise contributions of video system.

the results of these experiments applied to our robot workcell. The measurements will produce bounds on the individual components of the error, which we will then sum to obtain $\epsilon_{\mathbf{vision}_{xy}}$. Finally, we will verify this aggregate error bound by comparing $\epsilon_{\mathbf{vision}_{xy}}$ against measured error values observed over a large number of random trials.

3.3 Random Noise

The first part of the proposed error model to be analyzed is dot center location uncertainty caused by the presence of noise in the CCD, transmission medium and processor electronics. Noise may be defined as any unwanted signal present in an optical or electronic system. The primary noise contributions in the camera CCD are thermal currents, photon shot noise, and preamplifier electronic noise. Further noise contributions are made by transmission of the signal along the video coaxial cable, and repeated signal conversions from analog to digital and back to analog. It is difficult to accurately isolate and characterize each of these sources of noise error independently, but some inferences can be drawn. We expect that the noise will be essentially Gaussian in nature, and that it will contribute a small amount of error to the measurement of the dot center in the image frame.

To characterize the noise contribution to dot location error the following experiment was performed: A black anodized aluminum plate (see Figure 5) had eighteen holes bored to a depth of 0.016 inch. The holes were painted white to provide high contrast dots against the black plate. The plate was placed in the camera field of view and one hundred pictures were taken over an interval of approximately twenty seconds. The dot (x, y) locations were found using the dot location method discussed in Section 3.1 and recorded for each picture. With no camera, plate or robot movement and no light intensity changes, any variability in the dot location can be attributed to noise from the camera preamplifier, video transmission lines and the video processing system.

The raw data were stored to disk and transferred to a work station for processing using Mathematica^{TM}. For each picture the (x, y) center location was computed for each dot. The (x, y) center locations were also computed after averaging consecutive frames in groups of 3, 5, and 7 exposures. For each data set average locations along with minimum, maximum, and standard deviations were computed. Comparison of the data sets (see Figures 6 and 7) showed that the minimum, maximum and standard deviations are inversely proportional to the number of exposures used for averaging. Averaging also reinforces the data's Gaussian nature.

After studying the improvement in standard deviation attained versus the amount of time required to gather additional camera frames, we decided to average five frames for all subsequent static dot locating trials. This yielded a standard deviation of 0.0015mm for each dot center location. We chose a 6σ error bound to represent the worst case location error due to noise in the video system, resulting in $\epsilon_{\mathbf{noise}} = \pm 0.009$mm.

3.4 Camera Pixel Effects

An error term related to the deterministic physical and electronic properties of the camera CCD and video processing will be called pixel quantization. A camera CCD develops a charge on each pixel related to the number of photons reflected by an object which strike that pixel. A pixel fully illuminated by reflected photons will develop some maximum charge, and a pixel with no illumination will develop some minimum

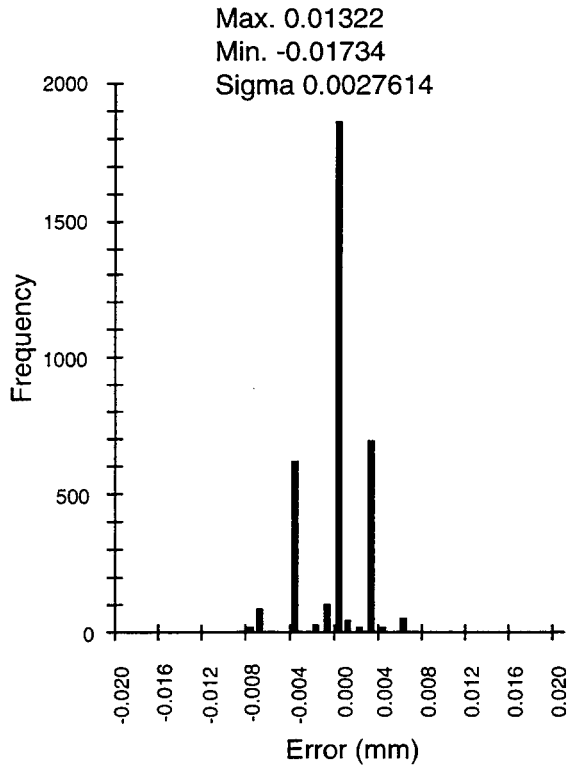


Figure 6: Histogram of raw data shows magnitude of hole location error attributable to electronic noise. Note the quantization effects, which we attribute to the fixed number of bits used to represent floating-point numbers in the Adept controller.

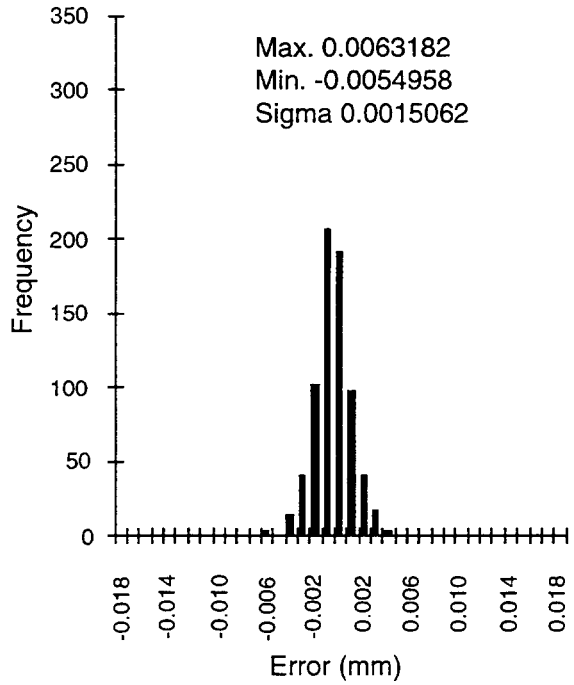


Figure 7: Histogram of averaged data shows magnitude of hole location error attributable to electronic noise. Each data point reflects the average of 5 hole center measurements. The improvement due to averaging suggests that the noise contribution is Gaussian.

charge. The pixels on the periphery of an object will be partially illuminated (see Figure 8), and develop a charge somewhere between these limits. Analog to digital conversion will quantize this charge to the nearest digital representation available. With infinite resolution the digital value will exactly represent the charge on the pixel. Given that infinite resolution is not possible (the video quantizer uses seven bits to describe the charge of each pixel) the digital representation is only an approximation of the illumination of the pixel. This approximation shifts the apparent location of the edge of an object at each point along the periphery of the object. Given that the dots have a diameter of 0.125in (3.175mm) and the pixel size is 0.40mm there will be approximately 50 pixels illuminated by each dot.

Depending on the location of the dot in the image, there may be as many as 14 pixels which are only partially illuminated. The actual number of partially illuminated pixels is unknown, as is the degree of illumination of each of these pixels. With a single axis translation of the dot locations, and uniform sized and spaced pixels, recurrent pixel illuminations should occur for each table movement equal to a pixel period. These assumptions led us to expect that the error in dot center locating ability of the video system would be periodic, with the period directly proportional to the pixel size.

Another contribution to pixel quantization is the inability of the video processing to exactly resolve the pixel into infinite divisions for the purpose of edge detection. Adept grayscale processing convolves a 3×3 matrix of pixel illuminations to determine image intensity gradients. The first partial derivative is then used to identify the location of the edge within the pixel. Tian and Huhns [12] report the error associated with this type of search as varying from 0.040 to 0.30 pixel, depending on scene parameters.

It is difficult to evaluate each of these sources and their contributions to camera error as individual quantities. We chose to characterize the effect of these camera errors as an envelope and evaluate their effect on the problem of registering image center locations accurately. This is analogous to the approach we used in characterizing noise errors.

The following experiment was run to evaluate the presence and effect of quantization error on dot center location accuracy. The grid used to study the effects of noise errors (Figure 5) was mounted on two sets of micrometer controlled tables at right angles to each other. We placed the grid in the image frame with its axes aligned to the vision frame axes.

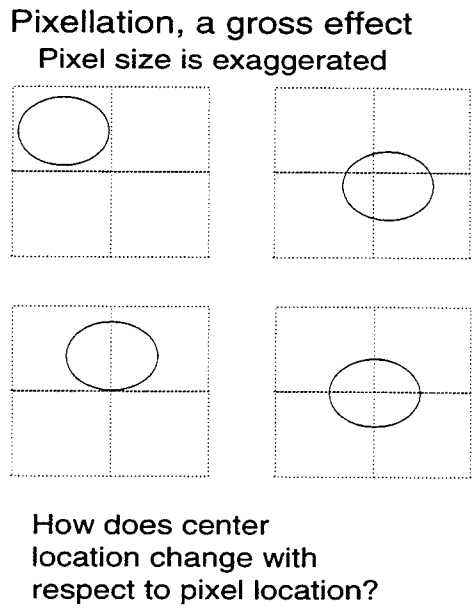


Figure 8: Illustration depicts the various places a pixel may lie with respect to the dot boundary.

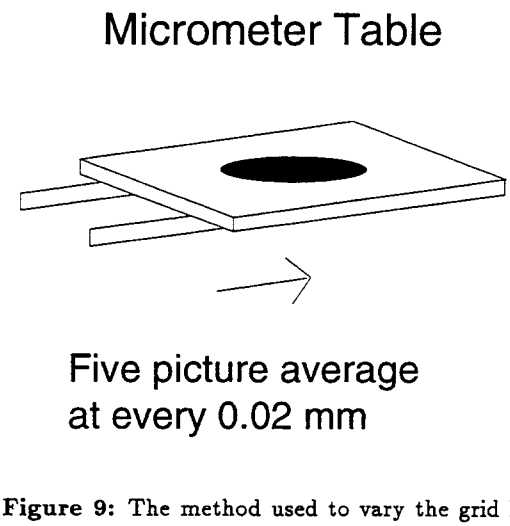


Figure 9: The method used to vary the grid location for pixel quantization measurement.

A baseline set of pictures was taken (5 exposures averaged per picture) and the dot center locations were found and stored to memory. We monotonically translated the grid's physical location in increments of 0.02mm in the x direction (see Figure 9) for a total movement of 2.0mm ($\approx$ 5 pixels). After each step the image frame locations of the dot center are found and stored to memory. This process is then repeated for displacements in the y axis. A plot was made of the location of each dot center with respect to the grid displacement along the x or y axis of the vision frame. The absence of quantization would yield a straight line plot where each dot center in the image frame is displaced exactly ($\pm\epsilon_{\text{noise}}$) the same distance as the grid is physically translated (Figure 10).

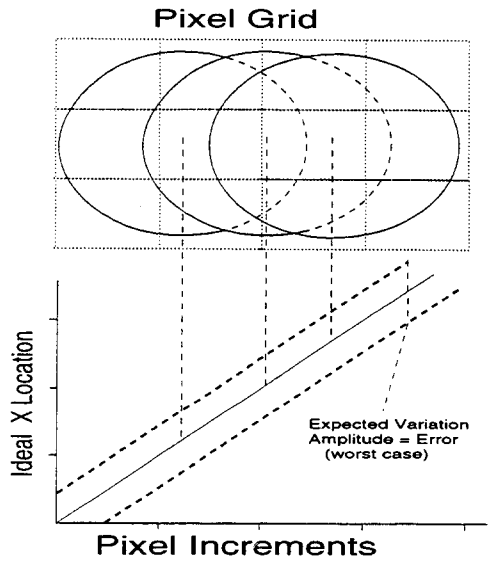


Figure 10: The solid line depicts the ideal response of measured dot locations with respect to plate movement. The dotted lines depict the expected envelope of variations due to quantization of pixel values in the image plane. These variations are expected to have a near sinusoidal distribution where the period is equal to one pixel size ($\approx$ 0.40mm), and an amplitude equal to the worstcase quantization error.

Refer to Figure 11 for the following discussion of the test results. Plots of the data versus the grid translation for each step yield responses typified by that shown in Figure 11. The sinusoidal response of the data has a period equal to the size of a pixel, as expected. To select an error envelope we drew lines parallel to the ideal response, through the maximum amplitude points of the sinusoid. The maximum amplitude observed in the plots is $\pm$0.060mm, or about $\frac{1}{8}$ pixel. This value is the maximum error that may result from the dot's particular placement relative to the grid of pixels. Thus $\epsilon_{\text{pixel}} = 0.060\text{mm}$.

3.5 Lens Distortion

Lens distortion is the failure of a point in the object plane to properly map into its correct location in the image plane. The major sources of lens distortion are driven by lens properties and physical imperfections of the camera.

There is a distortion proportional to the angular difference between a reflected light ray and the optical axis of the camera lens. This distortion causes object points to map into the image plane with some displace-

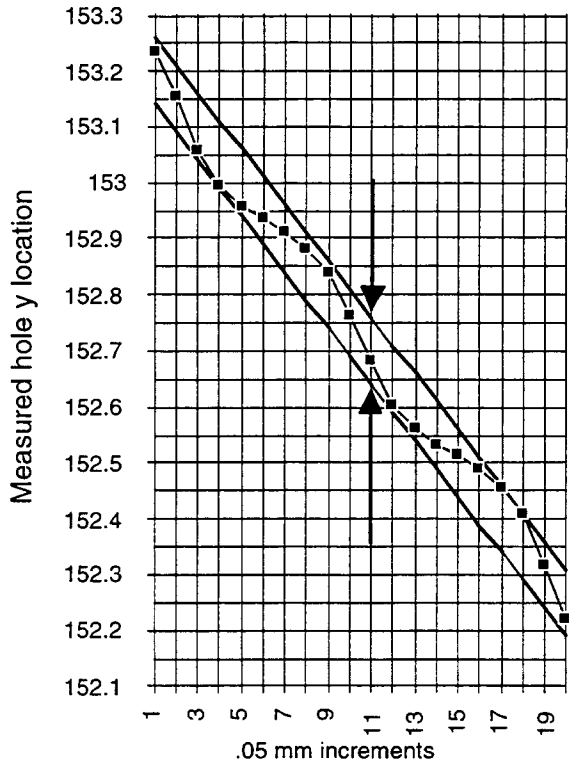


Figure 11: Graph showing actual deviations from ideal center location caused by pixel quantization effects. The worst-case error is ± 0.060mm, as shown by the arrows.

ment from their true location in the object plane. Negative values of the angle cause shrinkage of the image space (barrel distortion), and positive values cause expansion of the image space (pin cushion distortion). This distortion is primarily driven by the lens shape (wide angle, telescopic etc.).

Tangential lens distortion displaces image points in a direction normal to radial lines from the center of the image resulting in a skewed image plane. Causes of tangential distortion include imperfect physical alignment of the camera components, imperfect camera mount to the robot quill, or a non-perpendicular quill.

Figure 12 shows a graphical representation of the change in image shape with respect to several types of distortion. Once again it is very difficult to independently isolate and characterize the individual sources of distortion error within a given system. Our characterization of error components distorting the image (lens imperfections, physical mounting problems, etc.) will be as a composite error, referred to as lens distortion for the body of this work.

Our workcell employs a wide angle lens for our camera to provide the required field of view (200mm $\times$ 200mm) in the presence of severe physical occlusions presented by the quill mount location. The force sensor and the gripper both provide constraints to long focal

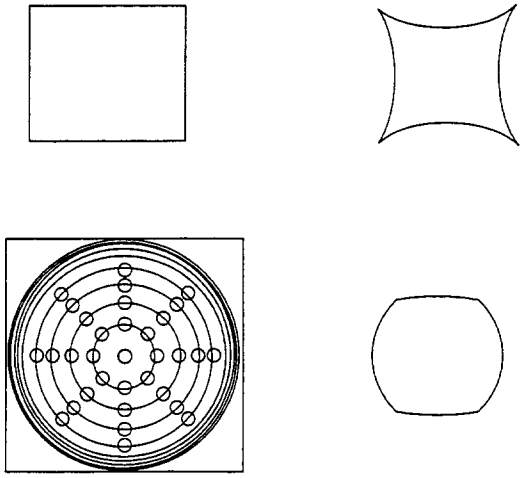


Figure 12: Some field of view alterations caused by various types of lens distortion.

length lenses which would need to be mounted further away from the object plane to provide the same field of view. To provide the required field of view, the camera lens must focus reflected light rays with significant angular displacement from the optical axis onto the camera CCD. Increased distortion of the image plane is a byproduct of the requirement to focus rays with significant angular displacement onto the CCD. The lens manufacturer's data sheet listed the lens distortion as approximately 4.5% of the distance from the optical axis to the point of interest. At a distance of 100mm from the image center, barrel distortion should be approximately 4.5mm. The magnitude of this error would drastically limit our ability to conduct experiments unless some sort of correction is applied to apparent image plane locations.

We characterized the image plane distortion using the following procedure: We placed a calibration grid with a 25×25 array of dots in the robot workcell. Each dot was 3.175mm in diameter, and milled to a depth of 0.406mm. The dots were spaced 10mm apart (see Figure 13).

This plate represents the object plane and is located at a focal distance of 280mm, which produces the desired 200mm $\times$ 200mm field of view. The camera (image frame) is centered over the grid (object frame) center point with a known location and orientation offset. This provides a known transform between the center of the image frame and the center of the object frame. Perfect physical camera element alignment and quill mounting would also produce an optical axis exactly perpendicular to the object plane. This would eliminate the need to take tangential distortion into consideration as a contributor to lens distortion. The inabil-

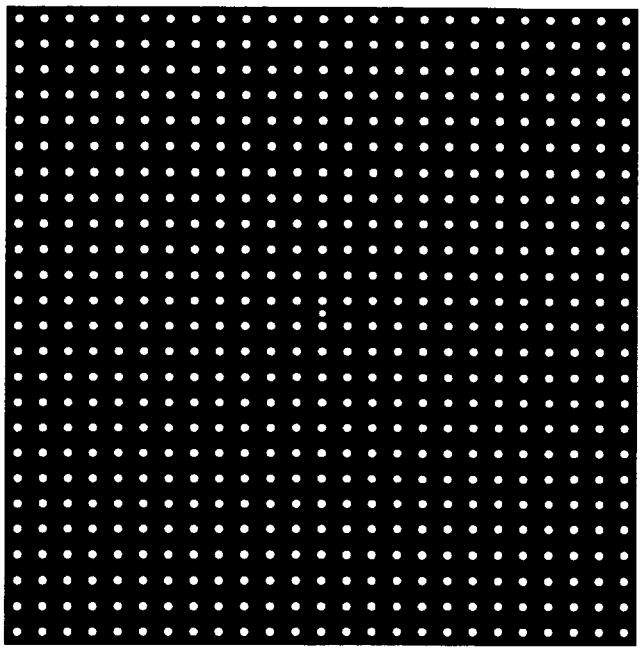


Figure 13: The 25 × 25 grid used to characterize lens distortion properties of the camera/lens. Note the fiducial mark used for angular alignment.

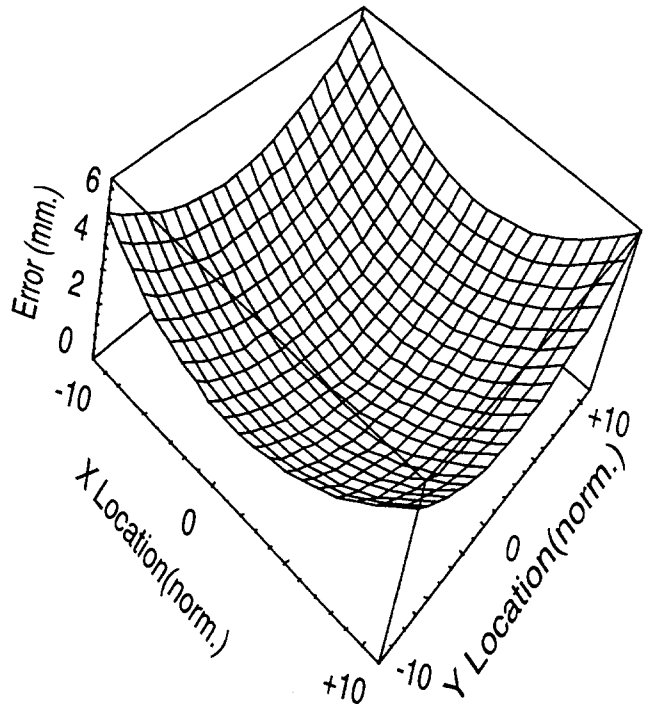


Figure 14: Error magnitudes of each hole center location before correction. The error magnitude is $\sqrt{(X_{\text{error}})^2 + (Y_{\text{error}})^2}$. Locations are referenced to the image frame (x, y) origin.

ity to perfectly align and orient the two frames requires measuring and saving the dx, dy, and $d\theta$ offsets as a transformation matrix to describe the relative position of the frames. A series of five pictures were taken and the dot center locations with respect to the grid center were computed and stored using the previously mentioned dot locating procedure. The dot center locations are transferred to a workstation for additional processing using Mathematica^{TM}.

The resulting data showed worst-case location errors of 4.98mm to 5.78mm (see Figure 14). The bowl shaped error surface clearly shows how the error magnitudes increase with distance from the image center. The non-symmetrical error distribution indicates that additional tangential error sources are combined with ordinary barrel distortion error. We rejected use of the approximate barrel distortion curve provided by the lens manufacturer based on these test results. We believe lookup tables based on the raw error characterization will enable a more accurate and comprehensive correction of all sources of lens distortion.

We collected the raw error values into two 21 × 21 matrices of error estimates, one for x_{errors} and one for y_{errors}. One way to describe these tables of error values is that each row describes the error as a function of x for a specific value of y, for each $y \in [1, 21]$. Likewise each column describes the error as a function of y for each $x \in [1, 21]$.

Several types of fitting routines were tested for accuracy in duplicating the error characteristics of the lens. A 3-d surface fit allows error estimate generation using a single equation. 3-d surface fits using $1st$ through $8th$ order equations for x and y errors yielded worst case error differences of 0.9mm at the gridpoints. In an attempt to obtain a more accurate representation, we evaluated other methods for fitting the data. Fitting the data row by row and column by column provided the best error replication. Because of fairly well behaved data for any given row or column, a very accurate fit was found. Fits for 3rd, 5th, 7th, and 9th order equations were evaluated to determine worst case errors. Error estimates were very good for all curvefits, getting better as the order increased (as expected).

Using numerous 2-d individual curvefits is more complex than using a single 3-d surface fit equation, but the increased accuracy makes this a worthwhile tradeoff. Once the curvefits have been found, each gridpoint is processed using the curves to generate error estimates. The error estimates are compared to the original error magnitudes to assess the quality of the curvefit (see Figure 16). We decided to use the 5th order fit because it represents a 50% improvement over the 3rd order, and higher orders provided asymptotically improved performance not worth the increased complexity. There are two sets of curvefit coefficients generated for each variable (x, y). One describes the error with

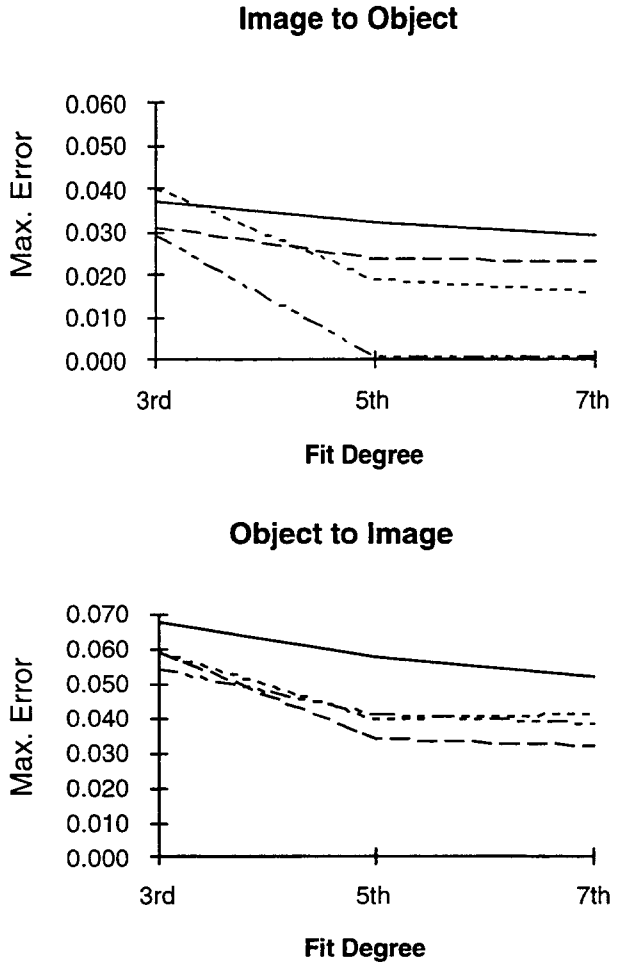


Figure 15: Line graphs show effect of increasing order of curvefits in attempt to model error behavior. There is one line for each curvefit function. Notice asymptotic improvement past 5th order for both graphs.

respect to a point's x location, and the other describes the error with respect to it's y location. We generated a set of datafiles containing the coefficients of the 5th order curvefit (a 21×6 array), and transferred these to the Adept controller. Because the curvefits are based on data taken at specific gridpoints, an interpolation method is required to assess error values at points of interest that do not lie on the grid.

3.6 Lens Error Correction

Correcting lens distortion errors in the video system is accomplished using lookup tables generated by the designated curvefits. There are two scenarios to be addressed: 1. For a point at a known location in the camera (image) frame, where is the point in the robot coordinate (object) frame? 2. For a known point in the robot coordinate frame, where should it be found in the camera image frame?

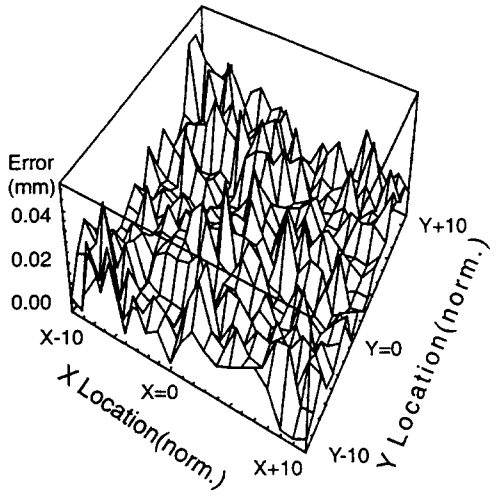


Figure 16: Error magnitudes of each hole center location after correction. Here the error magnitude is $\epsilon_{\text{measured}} - \epsilon_{\text{curvefit}}$. Locations are referenced to the image frame (x, y) origin.

The first step in correcting a location in either direction is the normalization of the location to the appropriate lookup table coordinate system (see Appendix A). This process assigns each point an index value signifying a relative (x, y) location in the current frame. If the normalization results in a fractional number, the point lies between characterized gridpoints, and interpolation will be required. The integer value of the assigned index (in both x and y), and the integer value plus one are used to select the set of curves required to calculate an error estimate. We now have the point location relative to the grid curve numbers of the error lookup table. For points which lie exactly on a gridpoint, the error can be directly calculated using the designated curvefits. The calculated error is added to the non-normalized (x, y) location to yield the corrected point location. Points which do not lie directly on a gridpoint are adjusted for errors using an additional set of curves and a 2-d interpolation to calculate the error magnitudes. The interpolation routine is discussed in Appendix A.

After implementing all of these interpolation methods, the worst-case residual distortion error can be estimated based on the deviation of observed data from the fit curves, and the additional error that may be introduced by the interpolation. This yielded a value of $\epsilon_{\text{lens}} = 0.05\text{mm}$.

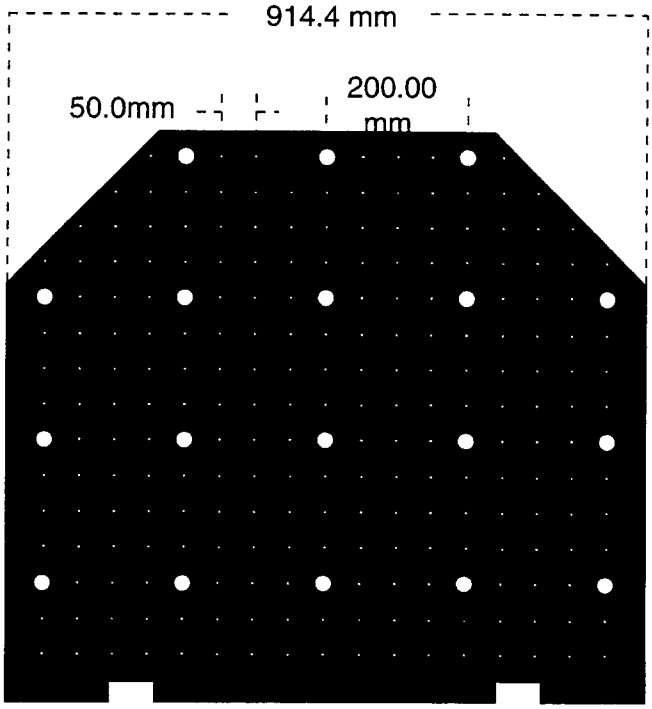


Figure 17: Grid used for checking vision error model. Triangles of various size and shape are produced by an overlay with holes cut in specific locations to allow dots to show through.

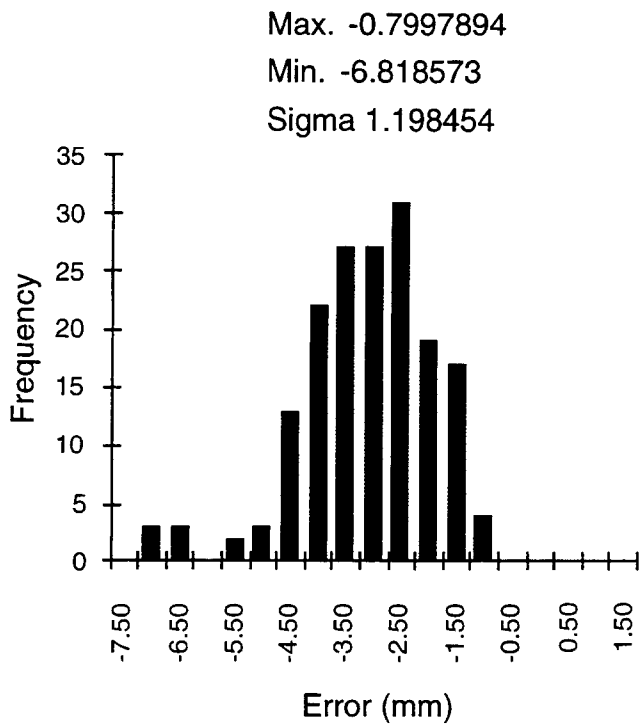


Figure 18: This histogram depicts the error in measured leg lengths, without distortion correction. These data reflect the measurement of 57 triangles, or 171 data points.

3.7 Testing Error Characterization

From the results of the previous sections we have

$$\begin{aligned} \epsilon_{\text{vision}_{xy}} &= \epsilon_{\text{noise}} + \epsilon_{\text{pixel}} + \epsilon_{\text{lens}} \\ &= 0.009\text{mm} + 0.06\text{mm} + 0.05\text{mm} \\ &= 0.119\text{mm}. \end{aligned}$$

It should be noted that the lens distortion error contribution after correction is less than the combined noise and pixel errors component. This is due to the quality of the curvefits employed to replicate lens distortion errors.

We devised an experiment to test the quality of the error characterization. A black anodized plate of dimensions 914.4mm × 808.48mm with a grid pattern of dots painted white is mounted in the robot workcell (see Figure 17). The grid is positively located by fixture balls mounted to the work surface allowing very accurate position repeatability of the grid in any of the four quadrants of the workcell. The grid spacing is 50mm in both the x and y directions, and the 0.875in dots occur every 200mm. The different dot sizes allow gross position identification based on dot size and facilitate extension of the calibration routines to the overhead cameras used in the workcell. Both dot sizes are used in the vision calibration and test routines of the quill camera. A set of locations was devised to provide image frame viewing of triangles defined by selected points on the plate grid. The triangles we selected represent a wide range of lengths and sizes, and were chosen to lie in various areas of the image frame as the manipulator moved among several test positions.

A cardboard overlay was made with one inch diameter holes punched out at specific locations, allowing only the dots of the desired triangle to show through. When the cardboard is overlaid on the grid, a series of triangles are visible to the camera from the selected locations. The robot visits the set of locations and records the image frame locations of the visible dots. Separations between the dots are calculated twice, once before distortion correction and once after. The calculated lengths are compared to the known leg lengths and the errors are stored in two files. See Figure 18 for precorrection errors, and Figure 19 for post-correction errors.

Note that for the raw data (see Figure 18) all error values are negative (image distances are shorter) with respect to the actual distance between the two points. This is a direct result of barrel distortion. The image field is compressed, and as an object moves further from the center of the image plane, the clustering effect becomes more pronounced.

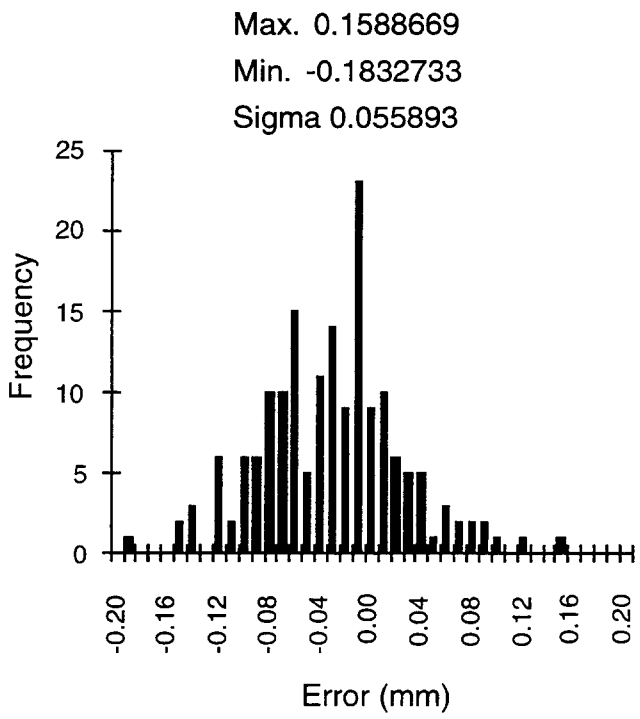


Figure 19: The error in measured leg lengths, with distortion correction. These measurements were made with the same images used in Figure 18.

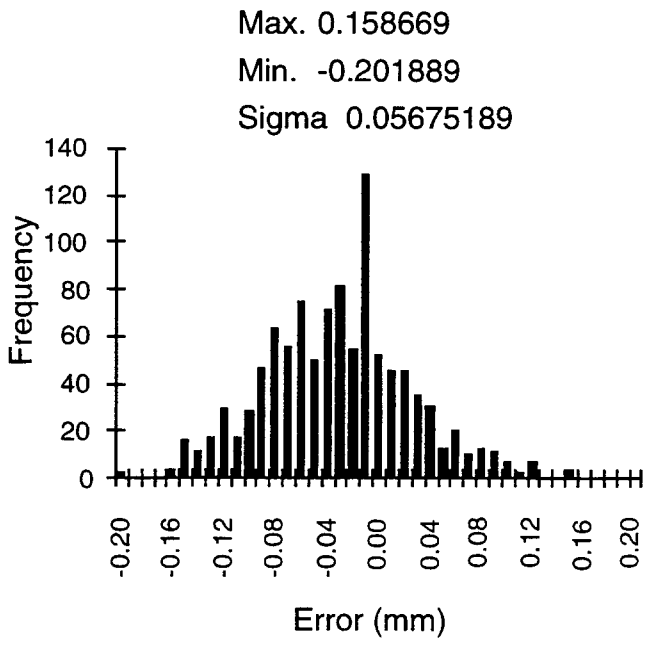


Figure 20: Error in measured leg lengths, with distortion correction. These data reflect the measurement of 347 triangles, or 1,041 data points.

3.8 Results

Figure 20 shows a histogram of 1,041 measurements used to test the validity of our worst-case error model. The largest error observed is less than two times $\epsilon_{\text{vision}_{xy}}$. Because this experiment measures two dot locations and calculates the distance between them, we would expect to see errors of up to but not exceeding $2\epsilon_{\text{vision}_{xy}}$. Thus these data support our estimate of the worst-case vision error.

4 Arm Calibration

4.1 Measuring ϵ_{motion}

We would like to apply the same approach to characterizing and correcting of robot kinematic errors that we used in the vision case: understand the process, identify constituent sources of error, and measure them in isolation. Unfortunately, we weren't able to clearly identify all such sources, and time did not permit an investigation that was as detailed as our vision investigation. Thus we applied a statistical method of measuring errors and post processing of data to quantify errors and apply corrections.

4.2 Absolute/Repeatable Positioning

The ability of a robot arm to position itself at some specified point in the workcell is affected by several factors. The resolution of the encoders used by each joint to decipher location information is one such factor. Differences between the kinematic model of the robot and the actual dimensions of the robot linkages and lengths, misalignment of the robot coordinate frame with an arbitrary world coordinate frame imposed by the user, friction properties of the robot joints, and variations in the robot operating environment are additional factors limiting the ability of the robot to be positioned with absolute accuracy.

It is important to distinguish between the absolute positioning capability and the position repeatability of a robot arm. Absolute positioning capability is the ability of the robot arm to go to a specified non-taught point in the workcell. It is directly related to variations between the kinematic model of the robot and actual physical dimensions of the individual robot. For example, the kinematic model of a robot depicts linkages as certain specific lengths even though linkages and link lengths vary from robot to robot due to manufacturing limitations. Most robot manufacturers do not specify absolute positioning accuracy due to the requirement to gather and analyze a large amount of data to certify a performance level for each robot manufactured. Typical robot systems depend on taught points or external sensing and feedback routines to minimize the requirement for this type of positioning accuracy.

It is critical to the typical robot user to be able to visit a specified point (taught or non-taught) with some known degree of consistency. This is called repeata-

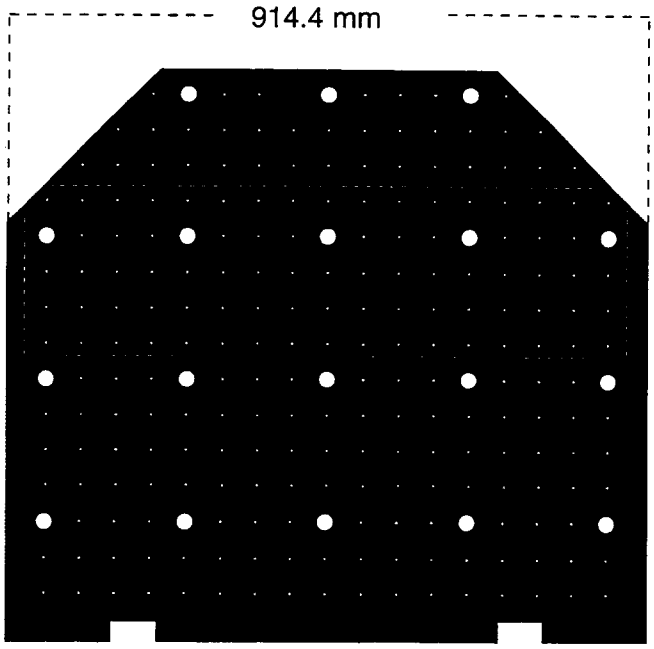


Figure 21: Grid used for checking absolute positioning error of robot. Area to be characterized is outlined by white box.

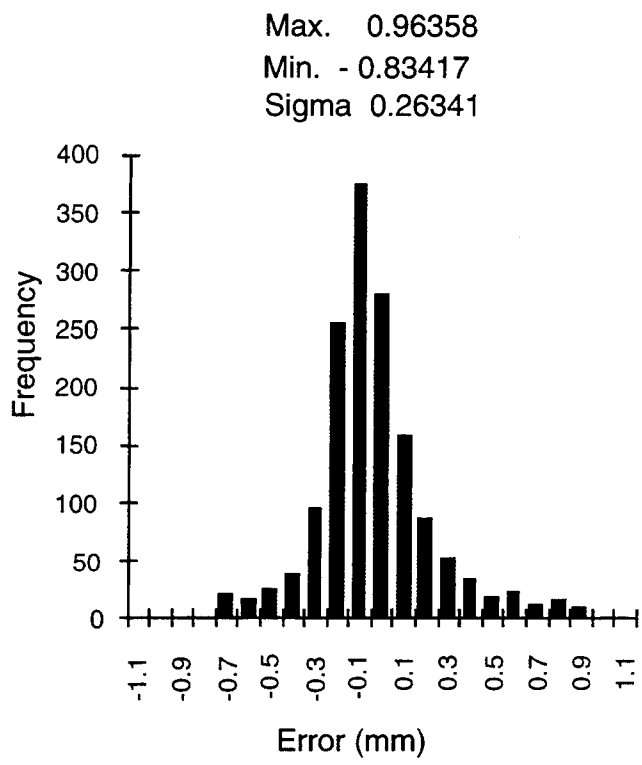


Figure 22: This histogram depicts the arm location errors compiled in three separate visits to the specified locations (in each quadrant). These data were used to develop the curvefits for each of the points visited in the workspace.

bility. Repeatability is affected by factors which vary less than those governing absolute positioning. Even though the physical parameters of a robot vary from the kinematic model, they are constant (excluding temperature effects) for a given robot. A manufacturer can calculate repeatability based on worst case manufacturing capabilities, electrical operating parameters etc. for a given class of robot. The resultant value determines the envelope of positions to which a robot, when commanded to visit a specified point, will go to every time. In the case of a robot commanded to visit a non-taught point, the resultant position may not be within the repeatability envelope of the commanded position.

4.3 Characterizating ϵ_{motion}

The grid used to generate triangles for checking the vision system errors will be used in this experiment. A rectangular subset (see Figure 21, area in rectangle) of all the points on the grid will be used to characterize a subset of the workcell. The grid may be positioned in each quadrant of the workcell facilitating arm position error characterization in a portion of each quadrant. The size of the corrected area is limited by several factors. To simplify curvefit routines the shape of the area must be square or rectangular, and the arm must be able to reach all points on the boundary of the area. An underlying assumption for this experiment is that the corrected camera error is smaller than the absolute positioning error of the robot. That is, we assume $\epsilon_{\text{vision}_{xy}} < \epsilon_{\text{motion}}$.

We will program the robot to visit points on the grid in three quadrants of the workcell using a software mapping routine with access to a matrix of stored world coordinate locations. The robot arm moves to a location and takes a picture of the grid. By locating multiple holes on the grid, the resulting location of the quill center after each move can be calculated (within the vision model uncertainty). The difference between the specified location and the resultant location is recorded as an offset $x_{\text{motion}_{\text{error}}}, y_{\text{motion}_{\text{error}}}$. The arm then visits each of the other grid points and repeats this process. The (x, y) location errors measured by the vision system at each point are mapped as a grid of error values. Completion of the routine results in the storing of the error grid to disk so it can be transferred to Mathematica^{TM} for post processing (see Figure 22). The same general curvefit routines from the lens distortion work were used to develop curvefits for the arm error data. The size of the rectangular arrays varied from those for the lens work, and varied with the workcell quadrant. For quadrants 1 and 3 the data was grouped into a 17×5 matrix, and for quadrant 2 data was grouped into a 5×17 matrix. The curvefit coefficients are stored in a set of data files and transferred back to the Adept controller.

4.4 Correcting ϵ_{motion}

To test the quality of the curvefits, we visited each point again. Before each move, the point to be visited was corrected using the curvefits as follows. The world coordinates of the desired point were analyzed to select the appropriate quadrant. The point was then normalized to the plate grid coordinates, and the error associated with that point was calculated using the curvefit equations. The error was then subtracted from the selected absolute point to provide an error corrected absolute location. We then commanded the robot to move to the corrected location. After the move the vision system was used to measure the arm's location. The offset from the desired absolute location was then calculated and compared to the pre-correction error magnitude for the same point.

The ability to correct arm position errors is quantified in Figures 22 and 23). Moving to the corrected location resulted in the robot achieving the commanded absolute position within an envelope of ± 0.130mm, yielding $\epsilon_{\text{motion}_{xy}} = 0.130mm$.

4.5 Results

From these data we conclude that our robot location error characterization, while not as detailed as the vision system characterization, does enable accurate corrections of the system errors. Note that the error corrected movements to absolute locations are accomplished with an error envelope only slightly larger than the corrected vision error.

Further testing was done on the quadrant 1 data because it displayed what appeared to be systematic error in the rotation offset calculated between the robot coordinate system and the user defined coordinate system (see Figure 24). This offset is required to establish a map from the robot coordinate system to the grid coordinate system prior to calculating absolute raw dot locations. The (x, y) error offset required to correct each data point to a zero rotation was calculated. All offsets were summed and averaged. The average was then subtracted from each raw data point, effectively minimizing the rotation offset. The new data was used to develop another set of curvefits for comparison with the original curvefits. The test results tracked the original curvefit errors almost exactly. One conclusion we draw from this comparison concerns the quality of the curvefit and correction routines. The curvefit is very good at predicting systematic errors, and the routines are robust in producing appropriate corrections to the errors.

In addition to running repeated correction trials, tests were also run vs. temperature from a range of approximately 60°F, to 85°F, which represents the extreme range of temperatures seen in our laboratory. The correction spread maintained a direct correlation to the distribution seen in the baseline tests, however the median value shifted with temperature. These data could be used to develop a temperature-corrected lookup table, but this remains for future work.

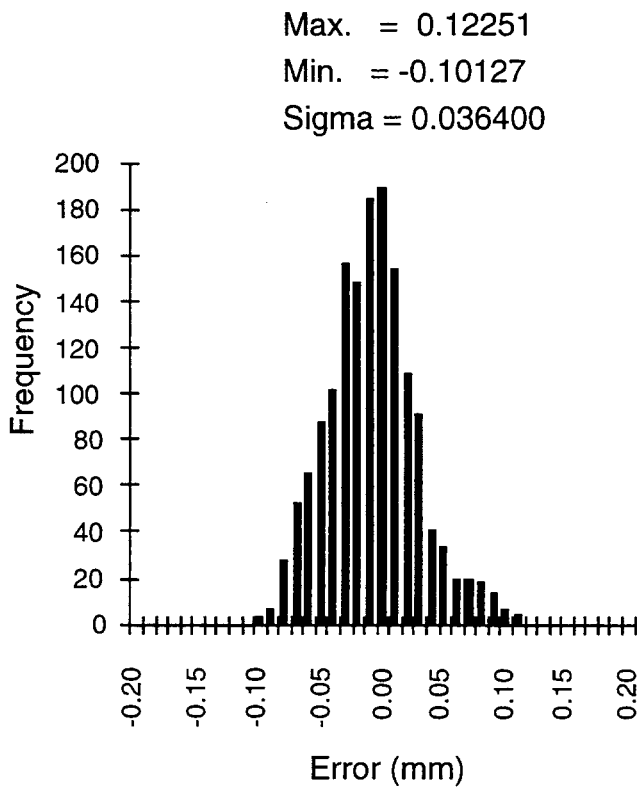


Figure 23: Arm motion errors, after correction.

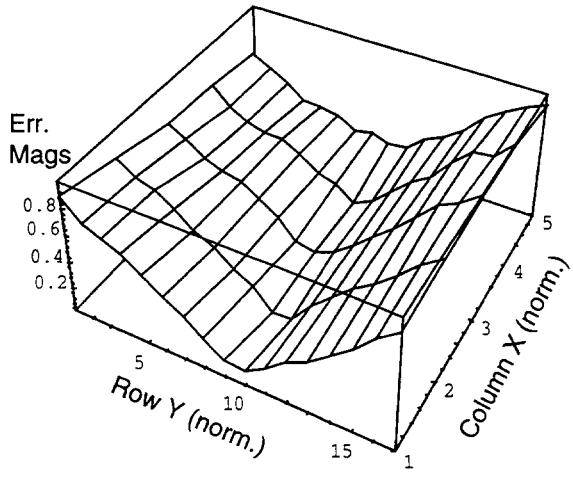


Figure 24: Graphical depiction of the arm position error magnitudes before correction. This quadrant is shown because it had the largest initial errors.

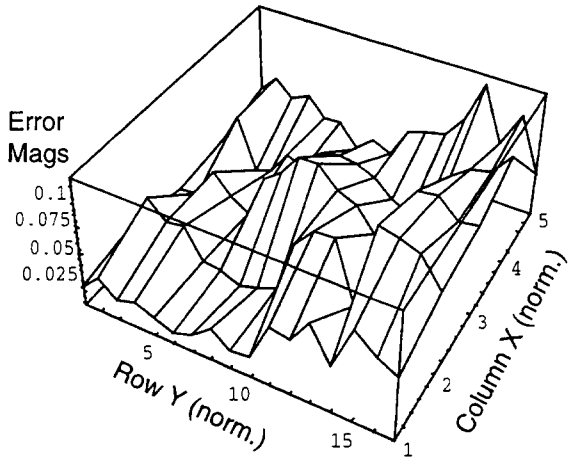


Figure 25: Graphical depiction of the arm position error magnitudes after correction.

5 Joint 4 Orientation Error

5.1 Error Characterization

To this point the error model of the robot arm consisted of error contributions of joints 1 and 2 only. We now turn our attention to the orientation of joint 4.

To characterize the motion accuracy of joint 4, a set of eight locations was visited by the arm. Each location was divided into 24 different orientations, one every 15 degrees. At each location and orientation the grid dots were measured visually to determine the actual orientation of joint 4. The difference between the intended orientation and the actual orientation was recorded as a joint 4 rotation error for that point. All locations were world locations, corrected via the lookup tables. After each measurement, the arm returned to a home location before proceeding to the next location and angle. This ensured that all points have approximately the same error contribution attributable to initial location error, and avoided stiction problems associated with motion through a small angle. The array of error values were then stored to disk and transferred to Mathematica^{TM} for further processing.

5.2 Results

Figures 26 and 27 show the results of this experiment. Evaluation of the error data from the eight holes shows that while the error magnitudes are consistant, the individual locations of the peaks and valleys are not. This denotes the need for individual curvefits based on evaluations of all individual areas that may be visited during the course of an experiment. We decided that given the magnitude and complexity of this requirement, the performance improvement attainable did not warrant the additional effort. Worst case error performance due to orientation errors will encompass the worst case error documented by the characterization of

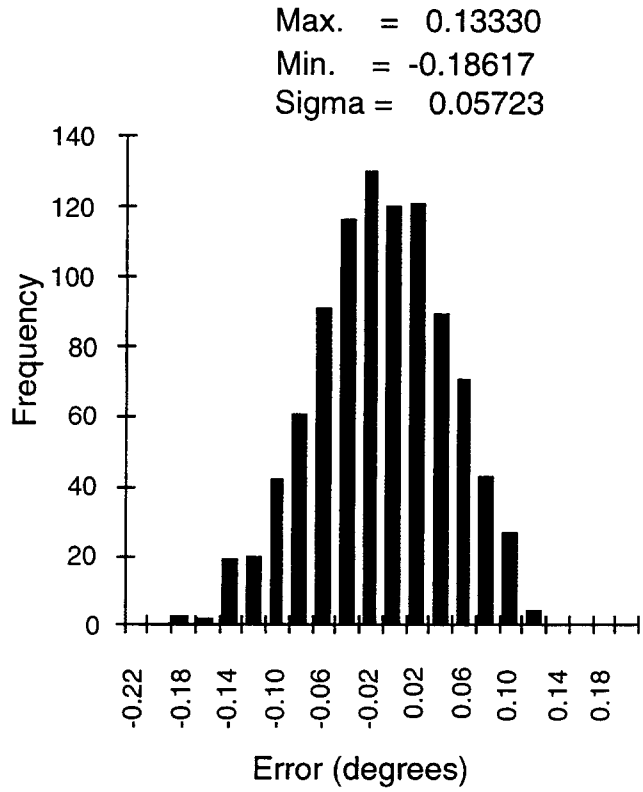


Figure 26: This histogram shows the results of visiting each of the eight holes at 24 different orientations with a step size of 15 degrees. The values depicted in the histogram are the error at each test point.

the eight holes. Thus $\epsilon_{\text{motion}_\theta} = \pm 0.20°$.

6 Summary

In this paper we successfully found error numbers that appear to capture worst-case error characteristics, and developed systematic methods for measuring and verifying worst-case error bounds. It is interesting to note that simply applying 6σ would have yielded poor results. To see why, consider the data shown in Figure 14 for these data, $\sigma = 1.1798$mm. The resulting 6σ bound would be ± 7.0788mm, which is ≈ 54 times the bound obtained through our experiments exploiting the nature of the physical sensing process.

One of the strong points of this work is the use of a common methodology for generation/correction of errors associated with widely variant processes. The same error generation/correction schemes are used for lens distortion and arm position correction. These methods may also be used to correct force sensing and finger positioning errors in future applications. The only variations on the scheme involve the different size

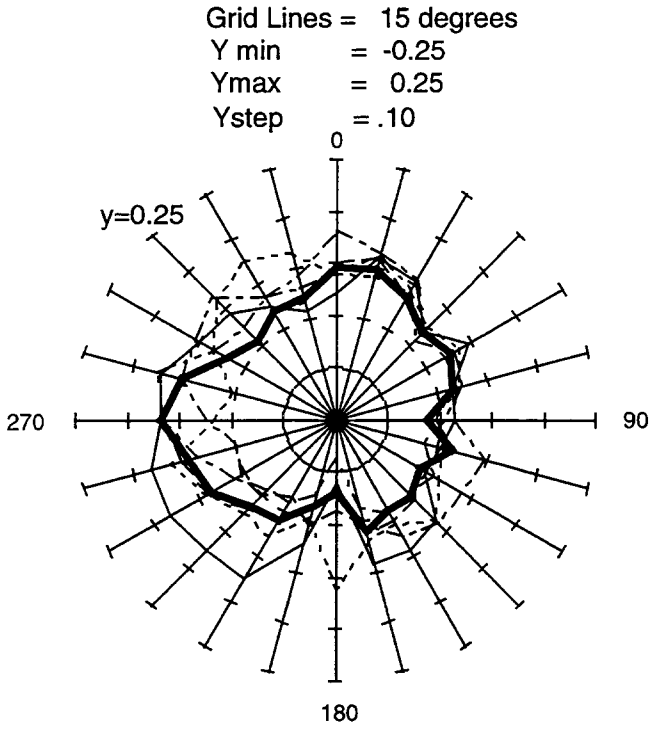


Figure 27: This circular plot compares the variation from each of five different sets of data for one hole. The bold line plots the average of the data sets. This hole is an example of the typical variations seen in plots of all the data points.

data files required to handle the coefficients of the curvefit equations as they vary by the size of the row column matrices.

On the down side, this calibration exercise required significant thought and work, even though the problems studied are ultimately pretty simple. This suggests that the work required to produce these characterizations for the myriad of parameters required to describe general manipulation tasks will be too costly to systematically apply in industrial practice.

Another problem is that as the space dimensionality increases, the required table size also increases. For this and other reasons, our methods become increasingly impractical as the task dimensionality increases. Thus it seems likely that prior arm kinematic identification schemes will provide better performance for full 3-d error estimation. When using these methods, how should we estimate worst-case error bounds?

Further, our initial idea of breaking errors down into their constituent components worked well in the vision case, but broke down in the arm case, leaving us with weak histogram-based methods. This is somewhat unsatisfying.

Since this is required for correct application of LMT worst-case analysis methods, we are led to conclude that this approach is impractical except in cases where extremely high reliability is required (e.g, assembly of inertial navigation instruments [14]). In the same vein, related work in our laboratory has concluded that the complexity of implementing planners that utilize these worst-case bounds is also onerous. Continuing work is examining the use of ordinary statistical error models for both planning and error characterization; these seem much more promising.

Appendix

Appendix A.1: Normalization

The grids describing error magnitudes are arranged so that the intersection of curve 11 in the x and y directions represents the grid origin (0,0). This point corresponds to the center of the image frame located at $(x_{\text{center}}, y_{\text{center}})$. To correct the position of a point, the point's (x, y) image location must be normalized with respect to the grid of interpolation curves. We accomplish this with the following equations, based on a 10mm grid spacing:

$$x_{\text{norm}} = \left(\frac{x_{\text{coordinate}} - x_{\text{center}}}{10\text{mm}}\right) + 11 \quad (1)$$

$$y_{\text{norm}} = \left(\frac{y_{\text{coordinate}} - y_{\text{center}}}{10\text{mm}}\right) + 11 \quad (2)$$

Appendix A.2: 2-d Interpolation

Interpolation will be used to correct errors in object and image frame locations which do not lie precisely (within ± 0.01mm in both x and y) on a characterized gridpoint. We tested several interpolation methods to identify the most accurate routine. We selected a bidirectional linear interpolation scheme to calculate the error magnitudes of points off the grid. Bidirectional correction makes use of data describing an error magnitude for each point, based on the point's (x, y) location. We believe this routine produces an accurate physical description of location errors caused by various sources of lens distortion.

The first interpolation is a weighted linear interpolation to provide an initial error estimate. A second interpolation will adjust this error estimate by adding a curvature correction. The curvature correction is significantly smaller in magnitude than the position error at all points of the image frame. The reason for this is that curvature over any ten millimeter segment connecting adjacent gridpoints is quite small — on the order of 0.02mm to 0.05mm mm. While an argument could be made for ignoring this error contribution $\epsilon_{\text{curvature}} \ll \epsilon_{\text{position}}$, our goal of high precision requires it's inclusion. The following example demonstrates the proccess used for finding a point's image plane location given an arbitrary object plane location.

Four curvefits are required to fully describe the error associated with any nongrid point. They are as follows:

1. x location error x_{error} as a function of x.
2. x location error x_{error} as a function of y.
3. y location error y_{error} as a function of x.
4. y location error y_{error} as a function of y.

The generation of these curves from observed data is described in Section 3.6.

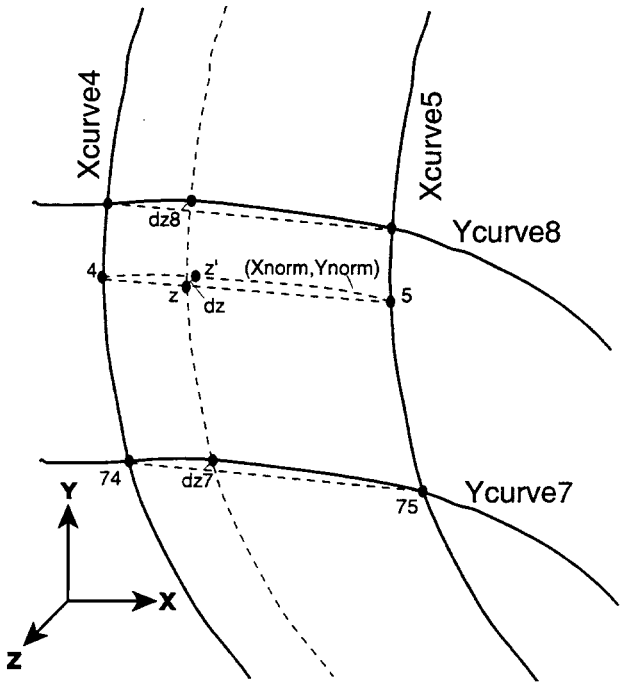


Figure 28: Graphical representation of the interpolation method used to generate error estimates for points of interest not located on specific gridpoints.

The following paragraphs will explain how this interpolation is performed for an example (x, y) point $(37, 64)$. Refer to Figure 28 for a graphical representation of the points that are generated.

Question: Given a dot located at point $(37, 64)$ in the image plane, where is the dot in the object plane?

Solution: We first normalize the point $(37, 64)$ to determine which set of curves to use for interpolation. Using equations (1) and (2) with values of $(x_{\text{center}}, y_{\text{center}}) = (104.8033\text{mm}, 100.7814\text{mm})$, we obtain $x_{\text{norm}} = 4.2167$ and $y_{\text{norm}} = 7.3219$. This shows that the point $(37, 64)$ lies between x curves 4 and 5, and between y curves 7 and 8.

Next we determine the correction values x_{error} and y_{error} through interpolation. We use the same interpolation method to determine both of these values, using the appropriate tables of x_{error} and y_{error} curvefit functions. In the discussion that follows, we will use the term z to refer to the x_{error} or y_{error} term that is being determined.

The first step in determining the value of z at $(x_{\text{norm}}, y_{\text{norm}})$ is to interpolate between the adjacent x curves. We will denote these curves $X_{\text{curve4}}(y)$ and $X_{\text{curve5}}(y)$; both of these curves are functions of y. Evaluating these functions at $y = y_{\text{norm}}$ yields $z_4 = X_{\text{curve4}}(y_{\text{norm}})$ and $z_5 = X_{\text{curve5}}(y_{\text{norm}})$, corresponding to the points labelled 4 and 5 in Figure 28. We then obtain our first estimate of the desired correc-

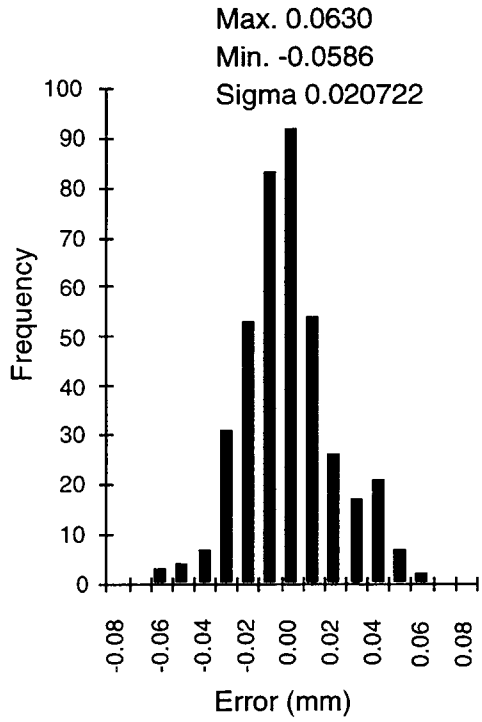


Figure 29: Dot location errors after single axis interpolation using a matrix of points that lie at worst case locations halfway between the curves.

tion value z by weighted linear interpolation:

$$z = z_4 + (x_{\text{norm}} - 4)\frac{z_5 - z_4}{5 - 4}$$

The resulting error estimate is reasonably accurate given the small amount of curvature present across the span of the boundary curves. Figure 29 shows the error magnitudes at worst-case points of interest using this single axis interpolation.

We can improve our estimate of z by using the y curves to estimate the surface curvature between X_{curve4} and X_{curve5}. Our goal is to compute a curvature correction dz which estimates the surface deviation from a straight line connecting points 4 and 5. We accomplish this by computing dz for y curves 7 and 8, and linearly interpolating between the resulting dz_7 and dz_8 values.

To obtain the value of dz_7, we first compute $z_{74} = Y_{\text{curve7}}(\lfloor x_{\text{norm}} \rfloor)$ and $z_{75} = Y_{\text{curve7}}(\lceil x_{\text{norm}} \rceil)$, corresponding to the points labelled 74 and 75 in Figure 28. We now calculate the linearly interpolated value

$$z_7 = z_{74} + (x_{\text{norm}} - 4)\frac{z_{75} - z_{74}}{5 - 4}$$

This value corresponds to a pure linear interpolation between points 74 and 75. We can now determine the curvature error for Y_{curve7} as

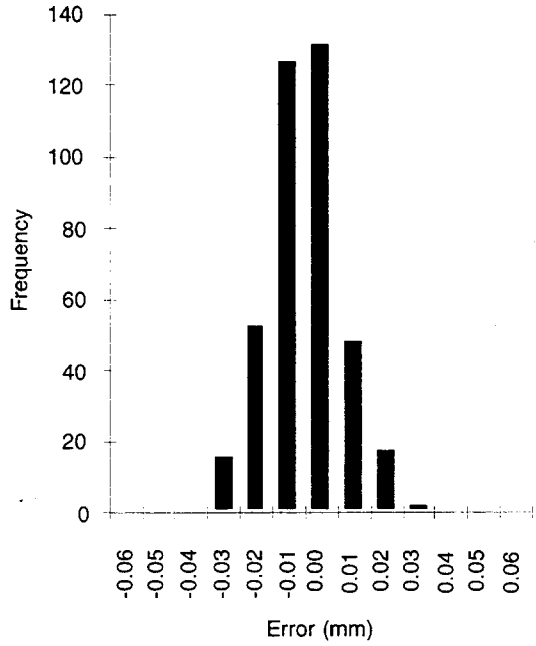


Figure 30: Dot location errors after double axis interpolation using matrix of points projected to lie at worst case locations. Note major difference in span from Figure 29.

$$dz_7 = Y_{\text{curve7}}(x_{\text{norm}}) - z_7$$

dz_7 reflects the surface deviation from linear at curve Y_{curve7}. We use a similar calculation to determine dz_8, the curvature correction at Y_{curve8}. We then estimate the curvature correction at $(x_{\text{norm}}, y_{\text{norm}})$ as

$$dz = dz_7 + (y_{\text{norm}} - 7)\frac{dz_8 - dz_7}{8 - 7}$$

We then obtain our final error estimate

$$z' = z + dz.$$

This method is used to find the correction terms x_{error} and y_{error}, using the appropriate curvefit lookup tables. These terms are then added to the point coordinates $(37, 64)$ to determine the corrected point in the image frame.

Figure 30 shows the error magnitudes at worst-case points using this bidirectional interpolation. The narrower range compared to Figure 29 corresponds to the improvement provided by curvature correction, which decreased the worst-case error by a factor of two. Note that the final worst case error estimate is well within the limits of the hole location error attributable to residual lens distortion errors.

Appendix A.3: Image To Object Frame Mapping

All previous work dealt with the problem of mapping object points to the correct image frame location. This helps us answer the question: Given a point on an object where should it show up in the image frame?

Another question we would like to be able to answer is: Given a point in the image frame, where is the corresponding point on the object? To answer this question we will require another set of error tables to map points from the image frame to the object frame. The error data to describe the offsets in this direction will be generated using the previous correction tables, a grid selected to provide convenient index locations, and an iterative routine for finding offsets between the two frames. When selecting the gridpoints and spacing to be used for the new error table, a couple of issues should be considered: First, for convenience the grid should be a 21 × 21 matrix of image locations, which allows us to re-use our curvefit and interpolation routines. Second, all values of the image-to-object error grid must lie within the boundaries of the previously defined object-to-image error grid. This requirement is driven by the use the object-to-image grid in an iterative manner to assign error values to image locations on the new grid. Locations outside of the previously found grid will cause the error estimation program to crash.

We will construct our image-to-object mapping by building x-curves and y-curves analogous to the object-to-image mapping. This is accomplished by visiting each image grid point (x_i, y_i), and numerically searching for the corresponding object point (x_o, y_o) which yields correction values $(x_{\text{error}}, y_{\text{error}})$ such that

$$\begin{aligned} x_i &= x_o + x_{\text{error}} \\ y_i &= y_o + y_{\text{error}} \end{aligned}$$

The correction terms associated with the image grid point (x_i, y_i) are then $-x_{\text{error}}$ and $-y_{\text{error}}$. For a given image grid point (x_i, y_i), this numerical search is accomplished using the following procedure:

1. Set our initial estimate of the object frame location to the given image grid point: $(x_o, y_o) \leftarrow (x_i, y_i)$.
2. Lookup the error correction for the current (x_o, y_o) using the existing object-to-image correction tables. This yields correction terms x_{error} and y_{error} for the current object point estimate (x_o, y_o).
3. Compute the image point corresponding to the current object point: $(x_i', y_i') \leftarrow (x_o, y_o) + (x_{\text{error}}, y_{\text{error}})$.
4. Compute the discrepancy in the image points $dx = x_i - x_i'$ and $dy = y_i - y_i'$.
5. If dx and dy are both less than ± 0.001mm, exit the loop and return $-x_{\text{error}}$ and $-y_{\text{error}}$ as the image-to-object correction values for (x_i, y_i). Otherwise, set $(x_o, y_o) \leftarrow (x_o, y_o) + (dx, dy)$ and go to Step 2.

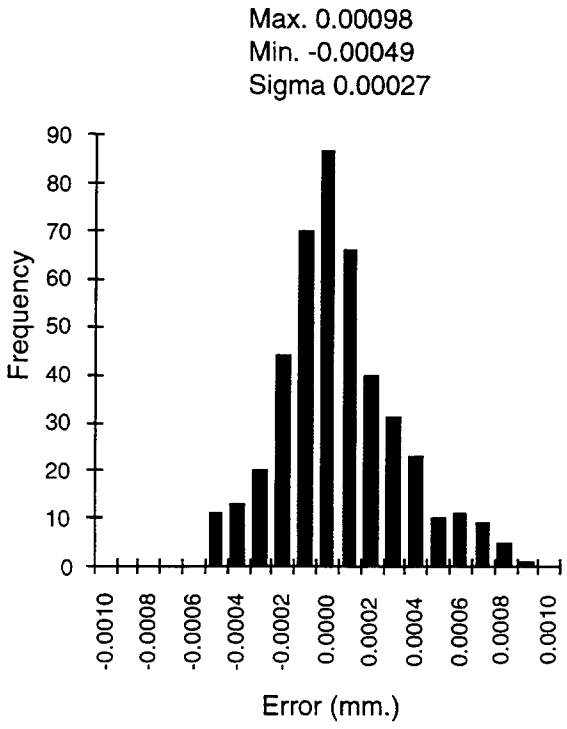


Figure 31: Histogram represents the difference between the actual image points (for all points on the calibration grid), and the error corrected image points.

Once correction values have been found for each image grid point (x_i, y_i), then 5th-order curve fitting and bidirectional interpolation may proceed as in the object-to-image case. Figure 31 shows the residual errors that remain at the end of this numerical search. All errors are less than the loop exit threshold of 0.001mm; most are significantly smaller.

We ran the following experiment to test both the interpolation routine and the newly developed error grid. Selected points in the object frame were corrected to find the image frame coordinates of the object location. After conversion, returned locations were used as object locations and input to the object to image correction routine. The locations returned from this routine were compared to the original image points for error offsets. The double conversion used points chosen to represent what we believed would represent worst case conditions for the interpolation schemes. The presumed worst case occurrence is a point halfway between two curves on both the x, and y axes, the point furthest from the curves used to characterize the worst case error offset. We collected data using an array of locations which represent the midpoints of all 21 curves. We also collected data using a set of 11 curves (every other curve). The location array for this data set consisted

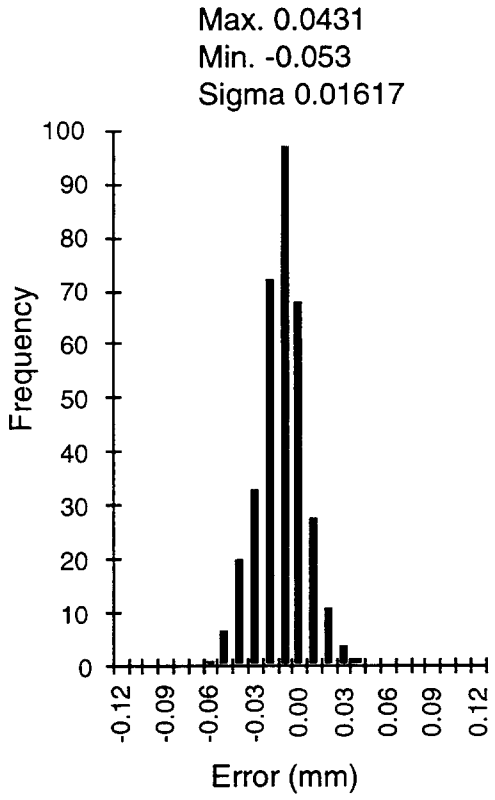


Figure 32: Graphical representation of the post interpolation errors using only 11 curves in each axis. This effectively doubles the distance from the characterized curves used for error estimations.

of points at which the unused x, y curves intersected. This allows comparison of interpolated data to true location data using a distance two times greater than will occur during normal interpolation calculations. Theoretically this should represent approximately twice the worstcase error magnitude. An error file of the (x, y) location errors after each test is stored to disk and analyzed using common statistics routines.

See Figures 32 and 33 for the following discussion. It is interesting to note the very small differences between the 11 curve and 21 curve error graphs. Lens distortion is very well behaved across the face of the lens. The lack of discontinuities results in replication of error results for both situations even though the distance from characterized curves is twice for the 11 curve test as for the 21 curve test.

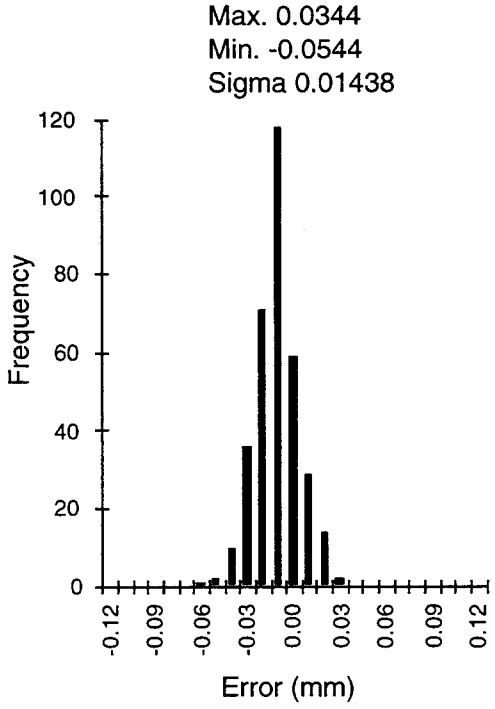


Figure 33: Graphical representation of the post interpolation errors using all 21 curves in each axis.

Appendix B.1: Motion Measurement Requirements

To begin the process of measuring arm position offsets several conditions must be assured. The camera height must reflect the proper focal distance (280mm) from the lens to the surface of the calibration plate. We wrote a software routine to identify the correct height in terms of z-coordinate parameters.

All absolute positioning will be done with respect to the quill center. This requires registration of the quill center in vision coordinates. A software routine rotates the quill and measures the location of a given dot (in vision coordinates) after each movement. A routine to fit an arc to the five points then locates the quill center in vision coordinates. Solution of the quill center in vision coordinates provides the transformation between the quill center and the image frame. This is the equivalent to finding the transformation from the robot coordinate system to the vision coordinate system using taught points (Adept's method). A significant advantage of our method is that the desired tooling is in place during the calibration process, meaning continuous checks and comparisons/corrections may be performed.

The approximate center of the test quadrant must be identified in an initialization program. This locates the quadrant to be calibrated and positions the robot to ensure the center of the calibration grid is located within the 200mm$\times$200mm window of the quill camera. Another software routine aligns the center of the image frame to the center of the calibration grid and stores the rotation matrix describing the offset between them.

References

[1] R. C. Brost. *Analysis and Planning of Planar Manipulation Tasks.* PhD thesis, Carnegie Mellon University School of Computer Science, January 1991. Available as Technical Report CMU-CS-91-149.

[2] D. C. Brown. Decentering distortion of lenses. *Photogrammetric Engineering*, 51:444–462, 1966.

[3] D. C. Brown. Close range camera calibration. *Photogrammetric Engineering*, 37:855–866, 1971.

[4] S. J. Buckley. *Planning and Teaching Compliant Motion Strategies.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, January 1987.

[5] M. A. Erdmann. On motion planning with uncertainty. Master's thesis, MIT Department of Electrical Engineering and Computer Science, August 1984.

[6] J. Latombe. Motion planning with uncertainty: On the preimage backchaining approach. In O. Khatib, J. J. Craig, and T. Lozano-Pérez, editors, *The Robotics Review I*, pages 53–70. MIT Press, Cambridge, Massachusetts, 1989.

[7] T. Lozano-Pérez, M. T. Mason, and R. H. Taylor. Automatic synthesis of fine-motion strategies for robots. *International Journal of Robotics Research*, 3(1):3–24, Spring 1984.

[8] P. Seitz. Optical superresolution using solid state cameras and digital signal processing. *Optical Engineering*, 27:535–540, 1988.

[9] Shishir Shah and J. K. Aggaral. A simple calibration procedure for fish eye (high distortion) lens camera. *IEEE Journal of Robotics and Automation*, 4:3422–3427, 1994.

[10] P.P. van der Smagt, F.C.A. Groen, and B.J.A. Krose. Robot hand-eye coordination using neural networks. Technical Report TR CS-93-10, University of Amsterdam, October 1993.

[11] Q. Tian and M. N. Huhns. A fast iterative hill-climbing algorithm for subpixel registration. In *Proceedings, 7th International Conference on Pattern Recognition*, pages 13–16, Jul/Aug 1984.

[12] Qi Tian and Michael N. Huhns. Algorithms for subpixel registration. In *Computer Vision, Graphics, and Image Processing*, pages 220–233, March 1986.

[13] R. Y. Tsai. A versatile camera calibration for high accuracy 3d vision metrology using off-the-shelf tv cameras. *IEEE Journal of Robotics and Automation*, 3(4):323–344, August 1987.

[14] D. E. Whitney. When people are too large and dirty. *Spectrum*, 30(9):39–42, September 1993.

Distribution:

MS 9018	Central Technical Files, 8940-2	(1)
MS 0899	Technical Library, 4916	(5)
MS 0619	Review & Approval Desk, 12690, for DOE/OSTI	(2)
MS 1002	P. J. Eicker, 9600	(1)
MS 1008	J. Fahrenholtz, 9621	(1)
MS 1008	R. W. Simon, 9621	(5)
MS 1008	R. R. Peters, 9621	(1)
MS 1008	S. Blauwkamp, 9621	(6)
MS 1008	T. Calton, 9621	(1)
MS 1010	D. Kholwadwala, 9622	(3)
MS 1380	Technology Transfer, 4212	(1)